IN THE CLAIMS

Please amend Claims 9 and 10 as follows.

SUB C17

1. (Original) A data structure, comprising:

in a heap tree or similar data structure, comprising:

a root level having a node group, the node group having k number of nodes; and

a second level having one supernode, the supernode having k number of node groups.

2. (Original) The data structure of Claim 1, further comprising one or more holes in arbitrary leaf positions, the one or more holes representing absent values.

3. (Original) The data structure of Claim 1, wherein the k number of node groups are siblings of each other in the heap tree such that only one sibling node is needed for any given path in the heap tree.

4. (Original) The data structure of Claim 1, wherein the arrangement of the supernode in the heap tree allows for speculatively reading a children node in the heap tree before an exact desired child node is known.

5. (Original) The data structure of Claim 4, wherein the determination of the exact desired child proceeds in parallel with the retrieval of the supernode.

6. (Original) The data structure of Claim 1, further comprising a third level having k number of supernodes.

7. (Original) The data structure of Claim 2, further comprising a remove or delete operation which does not require a last value to be moved into a root node.

8. (Original) The data structure of Claim 7, wherein the remove or delete operation comprises:

removing the value from the root node; and

percolating the hole associated with the root node down the heap.

9. (Currently Amended) The data structure ~~and remove operation~~ of Claim 7 [[2]], wherein the data structure contains a hole counter that counts the number of holes below the pointer for one or more of the pointers, the hole counter being associated with one or more pointers, the hole counter representing the number of holes in a sub-heap below the one or more pointers.

10. (Currently Amended) The data structure ~~restructure~~ of Claim 8, wherein the remove operation comprises incrementing by one the hole counter associated with each pointer that is traversed.

11. (Original) The data structure of Claim 2, further comprising an insert operation for percolating a value to be inserted starting at the root level and proceeding towards the bottom level.

12. (Original) The data structure of Claim 10, wherein an insert operation comprises:

percolating a value to be inserted starting at the root level;

in the one or more pointers, each pointer being associated with a hole counter that tracks the number of available holes, percolating the add value down a node in which the hole counter contains a value greater than zero; and

decrementing the selected hole counter by one.